JAVA ¿Un refugio para los expertos?

Ismael García Varea
José Hernández Orallo

En este artículo se comentan, desde un punto de vista crítico, los motivos para la aparición del nuevo lenguaje de programación Java y sus repercusiones en el mundo de la programación.

Respecto a las causas, ¿estamos una vez más ante el hecho de que una nueva tecnología se haya impuesto por sus intereses comerciales por encima de sus beneficios estrictamente técnicos?

Y una vez que tenemos Java por todas partes, ¿qué aplicaciones puede tener en Internet y fuera de ella?, ¿cuál va a ser su posición dentro de la gran rivalidad entre los lenguajes de programación? y, finalmente, ¿puede ser un área donde todavía sea necesario el uso de expertos?

EL FUTURO DE LOS PROGRAMADORES EXPERTOS

Si existe un lema que puede explicar la neurosis y estrés en la que se mueven muchos de los profesionales de la informática, podría ser más o menos el siguiente: "*el primer campo donde la informática intenta automatizar los procesos al máximo es, sencillamente, la propia informática*".

Bajo este prisma, es mucho más sencillo entender el desenlace de dos fenómenos fundamentales que se han producido en el mundo de la informática en los últimos cinco años:

En primer lugar, se ha producido un cambio radical en el mundo de los lenguajes de programación. Si el C++, consolidado a principios de los 90, constituyó un hito en capacidad, flexibilidad y popularidad dentro de la evolución de los lenguajes de programación, también lo hizo en complejidad. La mayoría de los jefes de proyecto, generalmente con menos conocimientos técnicos que los programadores, no se sentían cómodos ante una serie de conceptos tales como sobrecarga de operadores, herencia múltiple, polimorfismo, espacios de nombres o *templates*, que el C++ o las nuevas generaciones de otros lenguajes de programación proporcionaban. Esto provocaba una difícil interacción entre los programadores y su jefe, ya que el software era ininteligible para éste, lo cual repercutía en la calidad y tiempo de realización de los proyectos.

Frente a esta complejidad, y con la necesidad de generar aplicaciones rápidas para entornos populares (Windows principalmente), Microsoft irrumpió con sus herramientas visuales y popula-

rizó el hoy omnipresente VisualBasic. Cualquier persona –y no necesariamente técnico informático– con un cursillo rápido y un entorno visual, era capaz de hacer aplicaciones de pequeña escala bastante impresionantes y, fundamentalmente, en unos plazos mucho menores. Incluso comenzaban a afectar a la realización de las aplicaciones a media y gran escala, a no ser que se acudiera a productos estándar realizados por multinacionales. El futuro de los programadores expertos, sobre todo en España, estaba en entredicho.

En segundo lugar, la espectacular expansión de Internet, parecía que auguraba un nuevo mercado para los informáticos. En los primeros momentos y, ante el desconocimiento general, eran muy valorados aquellos informáticos que eran capaces de montar un servidor Web, establecer la conexión, y sobre todo, mantener el conjunto funcionando ante los inquietos usuarios de Internet. El clan de los *webmasters* había nacido. Pero pronto, y recordando nuestro lema, aparecieron paquetes integrados incluyendo todos los servicios de Internet en los que la instalación era cómoda, el mantenimiento prácticamente automático y los requerimientos hardware cada vez más baratos. Recurrir a un técnico para montar un servidor de web ya no era necesario ni rentable.

La situación de depreciación profesional de los ingenieros técnicos en informática era tal que muchos de ellos acababan haciendo páginas web (una tarea con mucha demanda y que inicialmente se pagaba bien), pero que hoy en día se está ya asignando (también gracias a ciertos editores semiautomáticos) a administrativos y maquetadores, con honorarios más modestos, o profesionales de la imagen y la publicidad, más indicados para este tipo de tareas si se desea un buen resultado.

Estos ejemplos son sólo los más significativos de un mundo, el de la informática, en el que la necesidad de servicios sigue creciendo a unos ritmos muy prometedores, involucrando más gente, pero de baja cualificación. Por contra, se ha estancado la demanda de los verdaderos profesionales, y muchos titulados deben aceptar trabajos muy por debajo de sus posibilidades.

El informático debe saber que si su trabajo se especializa demasiado puede correr el riesgo de no poderse adaptar a las nuevas necesidades, cada vez menos técnicas, donde priman el análisis, la integración, sintonización, optimización, la visión comercial, etc., aspectos que, dado su carácter inteligente, son más difíciles de automatizar.

Ante esta situación, no podemos volver a echar las campanas al vuelo ante la aparición de un nuevo lenguaje de programación. Por ello, antes de poder hacer un juicio, veamos qué es Java y qué viene a aportar, para saber quiénes y cuántos programarán con él.

¿QUÉ ES JAVA?

Java es un lenguaje de programación interpretado destinado principalmente a la interactividad y la interconectividad de entornos de red distribuidos y heterogéneos.

Con este propósito, en 1991, Sun pensó en el C++, pero este lenguaje no se ajustaba bien a una red distribuida, sobre todo en portabilidad y seguridad. Ideó un nuevo lenguaje, inicialmente llamado Oak, que a finales de 1992, se empezó a utilizar para control remoto. El proyecto quedó aparcado ligeramente hasta 1994, cuando la popularización del WWW hizo ver muchas posibilidades al lenguaje, que pasó a llamarse Java. La primera aplicación fue construir un navegador, que se denominó WebRunner y que después conoceríamos como HotJava.[1]

En realidad, Java es un dialecto simplificado de C++, al que se le han eliminado las partes más espinosas del lenguaje, tanto en complejidad como en fiabilidad (punteros, herencia múltiple, *templates* y sobrecarga de operadores), y se le han añadido otras nuevas, como el manejo dinámico de la memoria mediante un recolector de basura (garbage collector). Del C++ conserva, como características más importantes, la sintaxis, la filosofía orientada a objetos, la encapsulación de la información en clases y el tratamiento de excepciones.

[1] Para más información sobre los orígenes de Java: http://www.sun/sunworldonline/swol-07-1995/swol-07-java.html

Existen además numerosas facilidades para la interacción con los protocolos de Internet (TCP/IP, HTTP, FTP) e incluso, el lenguaje viene acompañado con una API (*Application Programming Interface*) estándar para la gestión de gráficos y multimedia. Aparecen librerías para la interacción con otros elementos, como la *Java DataBase Connectivity* (JDBC), aceptada como estándar desde su aparición.

Los programas en Java (cuya extensión habitual es .java) se compilan en un código intermedio denominado *compiled bytecode*, (organizándose en clases cuya extensión habitual es .class). Estas clases en código intermedio se interpretan posteriormente en una máquina virtual (algo así como un microprocesador emulado por software), que funciona utilizando el sistema operativo y el hardware de la máquina subyacente.

Tanto es así, que una máquina virtual Java consta de los mismos elementos que un procesador: un juego de instrucciones, un conjunto de registros, una pila, además de un recolector de basura y un área de métodos. Además la máquina permite la ejecución de varios hilos (*multithreaded*) de forma simultánea. La diferencia fundamental es que, al no ser un verdadero microprocesador, el *bytecode* se interpreta y no se ejecutan las instrucciones del computador directamente. La estructura de esta máquina proporciona dos cualidades muy importantes: portabilidad y seguridad.

En primer lugar, la máquina virtual puede realizarse en poco más de 40 Kbytes por lo que es posible la portabilidad a cualquier otro tipo de hardware. Además, las especificaciones de la máquina las proporciona Sun de forma gratuita, por lo que la mayoría de las firmas que se dedican a hacer compiladores tuvo prototipos de máquinas Java en pocas semanas.

En segundo lugar, el propio diseño de la máquina virtual mantiene multitud de verificaciones en el momento de la interpretación; no es posible desbordar un *array*, salirse del marco gráfico asignado, desbordar la memoria o asignar variables de distinto tipo. También posee un chequeo de integridad en el propio *bytecode* para asegurar que no ha sido manipulado incorrecta o malintencionadamente después de la compilación, por lo que el código es seguro y libre de virus. Como la mayoría de las verificaciones se realizan sobre el código, también incluye la capacidad de encriptar parte de los datos, para casos donde sea necesaria su privacidad.

Pero lo que ha catapultado a Java ha sido el hecho de que muchas empresas de software comenzaron también a incluir Java en sus navegadores web. Por medio de las *JavaApplets* se puede ejecutar Java justo en el momento de visualizar las páginas HTML, dotándolas de animación e interactividad, como si se tratara de cualquier aplicación local en entorno gráfico. Así, Netscape incluyó *JavaScript* en el Netscape Navigator 2.0 (un lenguaje de *scripts* inspirado en Java) en el primer trimestre del 96. Las versiones siguientes de éste y la mayoría de navegadores incluyen tanto JavaApplets como JavaScripts.

FUNCIONAMIENTO E INTEGRACIÓN CON INTERNET

Independientemente de Internet, y aunque sea interpretado, Java permite realizar *aplicaciones* de propósito general, muy parecido a lo que se puede hacer en Visual Basic, con la ventaja de que podrá funcionar en cualquier entorno. De todas maneras, actualmente éste no es el uso principal que se le está dando.

La mayoría del código que se está realizando con Java consiste en pequeñas *applets* que se ligan a una página web como se hace con cualquier otro recurso. En el momento de visualizar la página, el applet se ejecuta en local. Esto permite animaciones e interactividad en las páginas web, lo cual, hasta ahora, se restringía al uso de CGI, que se ejecutan en el servidor.

El código de los CGI reside en el servidor y no hay problemas de incompatibilidad con la interminable variedad de máquinas que pueden acceder como clientes del servicio. Con los CGI es posible implementar formularios y consultas a bases de datos, cuyo mecanismo es generar páginas al vuelo, es decir, dependiendo de cierta petición del usuario (consulta, búsqueda o selección) se construye una nueva página que es la que se envía para que se visualice. Este sistema tiene poca flexibilidad, pero se ha venido utili-

zando tradicionalmente porque no había otro hasta la aparición de Java.

Las applets, al ejecutarse en local, permiten usar todas las características de la máquina del usuario, sonido, gráficos, componentes del GUI (*Graphic User Interface*), tales como botones, combos, etiquetas, listas, *scrollbars*, etc., lo que permite efectos multimedia en la navegación por el WWW. Para mantener la seguridad e integridad de nuestra máquina de usuario, el navegador restringe el acceso a sus recursos (porción de la pantalla, periféricos, ficheros, etc.), restricciones que las verificaciones de la máquina virtual se encargan de preservar. El control de una applet se regula mediante eventos, tales como pulsaciones de teclas, movimientos de ratón, etc., dando el último toque de interactividad al conjunto.

El enlace entre el documento HTML y la applet se realiza mediante una etiqueta del tipo <APPLET Class ... >. Ésta referencia a una applet compilada, que puede estar ubicada en la propia máquina local, en el mismo servidor donde se obtuvo el documento HTML que se está visualizando o en cualquier otra dirección de Internet. Esto es similar a lo que ocurre con las imágenes incluidas en una página web que pueden estar localizadas en cualquier URL (*Uniform Resource Locator*).

Además de aplicaciones puras y de applets, en el entorno de Internet, Java permite realizar otros tres tipos de manipuladores de protocolo, manipuladores de contenido o métodos nativos. Los manipuladores de protocolo permiten tratar nuevos tipos de URL, como el ftp://, file://, http://, mailto://, etc. Por ejemplo, una dirección de un nuevo protocolo de telefonía que se llame *tele*, quedaría tele://www.upv.es. Los manipuladores de contenido son la contrapartida a los *plug-ins* de Netscape. Permiten tratar nuevos tipos de ficheros según la extensión de los mismos. Por ejemplo, nuestro navegador podría tratar ficheros que acabasen en .xls y mostrar la hoja de cálculo correspondiente usando Java. Por último, los métodos nativos permiten interrelacionar Java con C o C++.

Para finalizar, aclararemos que los JavaScripts, contrariamente a lo que parece indicar su nombre, no tienen demasiada relación con Java. JavaScript es un lenguaje de macros o scripts, con un chequeo de tipos bastante débil, con una sintaxis también inspirada en el C/C++ y con unas capacidades limitadas y muy orientadas a completar el HTML. El hecho de que Java y JavaScript sean dialectos del C++, hace que parezca que estén íntimamente relacionados. Pero en realidad, cuando Netscape decidió incorporar un lenguaje de scripts en su Navigator 2.0, inicialmente se iba a llamar LiveScript, pero por motivos comerciales, Netscape obtuvo la licencia de Sun para darle el nombre definitivo de JavaScript.

Hasta aquí hemos visto una breve introducción a Java y su interrelación con Internet. Si se desea más información sobre el lenguaje en sí y todas sus posibilidades, la facultad de informática y estadística de Sevilla ha puesto en http://www.fie.us.es/info/internet/JAVA un tutorial de Java en castellano. Para profundizar en el tema o ver algunos ejemplos de JavaApplets y JavaScripts, lo mejor es comenzar en http://www.javasoft.com o http://www.sun.com.

¿POR QUÉ JAVA?

Hay varias características que parecen marcar la aparición y el éxito del fenómeno Java:

- Los servidores de HTTP estaban saturándose de CGI. Por el contrario, el paradigma Java utiliza la CPU del usuario, desplazando la inteligencia (y la carga) del servidor al cliente.
- Esto permite el uso flexible de recursos de la propia máquina del usuario, principalmente multimedia.
- Como resultado, aumenta la interactividad del web con el usuario, hasta ahora limitada a ir de unas páginas a otras.
- Las API que acompañan a Java están enfocadas a Internet.
- Además de las API, aparecen cada día más librerías estándar, como las JDBC (*Java DataBase Connectivity*).

- El hecho de ser multiplataforma, permite la distribución e integración de software.
- Facilidades como la documentación automática o los Java Beans agilizan todavía más la reutilización.

Pero no es oro todo lo que reluce:

- Java no es tan sencillo de usar ni de aprender.
- Java es interpretado, lo que lo hace lento para muchas aplicaciones.
- Las applets pueden tardar varios segundos en telecargarse por Internet, haciendo que los CGI sean más idóneos en algunos casos.
- Java no es completamente portable y la apariencia de los componentes GUI depende del sistema operativo local.
- Java no es apropiado para hacer grandes proyectos.

Además, Microsoft tenía planes para colocar a su omnipresente VisualBasic en el lugar que hoy en día ocupa Java. De hecho, ha realizado un VisualBasicScript para rivalizar con JavaScript (aunque ambos van incluidos en el Explorer). Pero la flamante contrapartida Microsoft a las clases Java son los inicialmente llamados OAO (*OLE Automation Objects*) y después renombrados a un más comercial "ActiveX". Los ActiveX están compilados para la arquitectura Intel, por lo que su ejecución es muy rápida, pero el problema es su portabilidad, ya que sólo funcionan en sistemas Intel-Windows.

Ante estas ventajas, inconvenientes y rivales de Java, uno se puede plantear por qué no se han adaptado otros lenguajes como VisualBasic, ObjectPascal o el mismo C++. Incluso existían ya lenguajes que siguen la misma filosofía que Java: Smalltalk tiene una sintaxis agradable, es interpretado, es tipado y orientado a objetos, tiene recolección automática de basura y existen espléndidas herramientas multiplataforma, como el *ParkPlace Smalltalk Environment*.

Pero influyen muchas otras razones para que una nueva idea se imponga, aunque no sea mejor que otras ya existentes.

En primer lugar, cada vez que aparece una novedad en informática, la industria de la formación se pone en marcha. Se crean cursos, conferencias y seminarios, se escriben libros o artículos como éste en revistas como ésta, para informar y reconvertir a los profesionales al nuevo paradigma.

En segundo lugar, la mayor parte de compañías del mundo de la informática han preferido que Java de Sun se impusiera sobre VisualBasic de Microsoft, ante la posibilidad de que esta última monopolizara también Internet. En particular, ha sido crucial la reacción del mundo Unix/C/C++, que ha visto cómo el VisualBasic y otras herramientas visuales le iba confinando únicamente a grandes proyectos de ingeniería. Tanto es así, que han visto en Java una posibilidad de resurgir, de que su experiencia y habilidades sigan siendo imprescindibles por algún tiempo más.

EL SUEÑO DE LA REUTILIZACIÓN DEL SOFTWARE

Supongamos que se superan los pequeños problemas actuales de rendimiento y portabilidad (diferente apariencia o comportamiento) de las API de Java y las JDBC. Java incide directamente en el desarrollo e instalación de programas.

Además, con la nueva arquitectura de componentes denominada Java Beans, se pretende crear una API neutral entre distintas plataformas, para facilitar la reutilización de componentes dinámicos Java. Sun Microsystems [10] espera que "Java Beans se podrá reusar en una variedad de herramientas tales como HotJava, Netscape Navigator, Borland's JBuilder, Symantec's Visual Café, IBM's Visual Age for Java, Microsoft's Internet Explorer, Visual Basic, Visual J++, Word, and Claris Works".

El ciclo de vida del software se puede simplificar mucho si analistas y diseñadores tienen acceso y conocimiento de las librerías de Applets y Beans Java ya existentes y hacen un buen diseño de la aplicación, teniendo en cuenta qué piezas, ya fiables y eficientes, tienen al alcance. El papel de la etapa de codificación (fundamental en la programación tradicional) se reduce drásticamente.

Además, gracias a Internet y la propia filosofía Java, las últimas etapas del ciclo, como la instalación remota y el telemantenimiento, se aceleran y simplifican, requiriendo menos técnicos y desplazamientos.

En este panorama, los componentes de los programas del futuro serán (más aún de lo que ya lo son hoy en día), construidos por diferentes desarrolladores en momentos distintos, un proceso que ha sido más tangible recientemente. Con los Java Beans, los componentes no se ensamblan entre sí sino que se intercomunican facilitando que las aplicaciones evolucionen a medida que progresan o se sustituyen sus componentes.

Además, la reutilización del software se refuerza con herramientas para la generación automática de documentación y su acceso instantáneo a través del WWW. El conjunto, es lo que se conoce como "Programación Hiperenlace Distribuida".

CONCLUSIONES

A primera vista, y como cualquier novedad, Java plantea una expectación desmesurada. Muchos profesionales de la informática ven en Java un lenguaje lo suficientemente complejo para que pueda mantenerles en una posición privilegiada dentro del mundo de la programación. De hecho, su figura se está viendo amenazada por aficionados menos cualificados, que realizan aplicaciones impresionantes en entornos visuales con relativa facilidad, y cuya mayor destreza reside en el manejo del ratón.

Este último intento de mantener o recobrar su *status* a toda costa, no sólo no es muy loable por su parte, sino que, a nuestro parecer, agravará todavía más su situación. El hecho de que el software se pueda reutilizar y obtener fácilmente en la Red, y la naturaleza altamente competitiva del mundo Internet, hace que pululen ya toneladas de software realizado en Java. En el futuro, la competencia de programadores de otros países en vías de desarrollo, con unos honorarios mucho menores, puede hacer que las esperanzas del resurgir de la programación tradicional no sean más que castillos en el aire.

REFERENCIAS

[1] Carlson, Bob, "A jolt of Java could shake up the computing community", *Computer* (Nov 1995), 81-82.

[2] Friendly, Lisa, "The Design of Distributed Hyperlinked Programming", Sun Microsystems Inc. 1995 (http://www.sun.com/../iwhd.ps).

[3] Gosling, James; Joy, Bill; Steele, Guy, "The Java Language Specification", Addison-Wesley, 1997.

[4] Hernández Orallo, Enrique; Hernández Orallo, José, "Programación en C++" 2ª edición, Editorial Paraninfo, 1995.

[5] Hernández Orallo, José; Talens Oliag, Sergio, "Internet. Redes de Computadores y Sistemas de Información", Editorial Paraninfo, 1997.

[6] Jaworski, Jamie, "Java Developer's Guide", Sams Net, 1996.

[7] Keaton, John; Hamilton, Scott, "Employment 2005: "Boom or Bust for Computer Professionals?", *Computer* (mayo 1996), 87-98.

[8] Lewis, Ted, "Will tiny beans conquer the world again?", *Computer* (septiembre 1996), 13-14.

[9] Shiffman, H., "Making Sense of Java", 1996-1997 (http://reality.sgi.com/employees/shiffman_engr/Java-QA.html).

[10] Sun Microsystems, "Java Beans: A Component Architecture for Java", diciembre 1996 (http:// splash.javasoft.com/beans/WhitePaper.html).

[11] Van Hoff, "Java and Internet Programming", *Dr. Dobbs Journal* (agosto 1995), pp. 56-61.

[12] VBPJ Staff, "What are Programmers Paid?", Visual Basic Programmer's Jorunal, febrero 1997, 84-94, también en (http://careerlink. windx.com).